Tuukka Mikola

Automatic Data Collecting from the Test Systems

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Electrical Engineering

Thesis

26 May 2016

Helsinki Metropolia University of Applied Sciences

Tekijä Otsikko Sivumäärä Aika	Tuukka Mikola Automaattinen datankeräys testausjärjestelmästä 21 sivua + 2 liitettä 26.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Sähkötekniikka
Suuntautumisvaihtoehto	Sähkövoimatekniikka
Ohjaajat	kehityspäällikkö Eero Saarela lehtori Eero Kupila

Tämä insinöörityö tehtiin ABB Oy:n Vantaan DSW-korjaamolle ja koko konsernin käyttöön. Projektissa suunniteltiin automaattista datan keräys järjestelmää, jota voitaisiin hyödyntää korjaamoverkoston testaus alueilla kaikkien ACS800 taajuusuuttajien lopputestauksessa. Datan tarkoituksena on parantaa loppuraportoinnin laatua ja huomata paremmin taajuusmuuttajissa olevia mahdollisia piilovikoja, kun dataa vertaillaan pidemmällä aikajaksolla.

Lähtökohtana oli vertailla NETA-21 ja ohjelmoitavan logiikan eroja, sekä toteuttaa järjestelmä parhaalla mahdollisella tavalla. Kuitenkin PLC oli jo alusta asti ensisijainen vaihtoehto ja työ toteutettiinkin ABB:n AC500-eCo PLC:llä. Projektin aikana järjestettiin muutamia tapaamisia ja pidettiin yhteyttä sähköpostilla ABB:n ammattihenkilöiden kanssa, joissa neuvoteltiin aikatauluista ja budjeteista, sekä projektin yksityiskohdista.

Insinöörityössä on selvennetty ohjelmoitavan logiikan, taajuusmuuttajan, NETA-21 ja Modbus perusteet sekä PLC:lle tehdyn ohjelman toimintaa. Lisäksi työssä kerrotaan mitä tietoja PLC:stä halutaan ohjelman valmistuttua. Ohjelmoitava logiikka ja sen ohjelmointi oli projektin painopiste. Ohjelmoitavan logiikan ohjelma on tarkoitettu pääsääntöisesti vain ABB:n omaan käyttöön.

Ohjelmoitavan logiikan ohjelmointiin käytettiin ABB:n omaa Codesys – ohjelmointiympäristön sisältävää tietokoneohjelmaa Automation Builder. Ohjelma osaa lukea taajuusmuuttajasta halutut parametriarvot ja tallentaa ne ohjelmoitavan logiikan muistiin. Tekstitiedoston kirjoittaminen tarvitsee vielä lisäkehitystä, jotta dataa saadaan ulos halutussa muodossa.

Avainsanat	AC500, ohjelmoitava logiikka kontrolleri, PLC, NETA-21, ACS800, taajuusmuuttaja, Modbus, ABB

Author Title Number of Pages Date	Tuukka Mikola Automatic Data Collecting from the Test Systems 21 pages + 2 appendices 26 May 2016
Degree	Bachelor of Engineering
Degree Programme	Electrical Engineering
Specialisation option	Electrical Power Engineering
Instructors	Eero Saarela, Development Manager Eero Kupila, Senior Lecturer

This thesis was carried out for ABB Ltd. DSW Finland workshop and the entire corporation. The project was to design an automatic data collection system, which could be utilized in the repair shop testing areas of all ACS800 frequency converters final testing. The purpose of the data is to improve the quality of the final reporting and increase the notification of possible hidden faults in frequency converters, when data is compared in the long run.

The starting point was to compare the NETA-21 and the programmable logic differences, and execute the system in the best possible way. The PLC had been the first alternative since the beginning and the work was carried out with ABB's AC500-eCo PLC. During the project a number of meetings were organized and contact was held with ABB professionals via email about schedules, budgets, as well as details of the project.

The basics of programmable logic, frequency converter, NETA-21 and Modbus have been clarified in the thesis, as well as the functions of the program made for the PLC. There is also information of the results wished to extract from the PLC once the program is complete. Programmable logic and its programming was the focus point of the project. The programmable logic program is mainly intended to be used internally at ABB.

For programmable logic programming, ABB’s own PC program called Automation Builder was used, which includes Codesys programming environment. The program is able to read the desired parameter values from the frequency converter and save them on the programmable logic’s memory. Writing a text file needs further development, so that data can be extracted in a desired format.

Keywords	AC500, Programmed Logic Controller, PLC, NETA-21, ACS800, Frequency converter, Modbus, ABB

Contents

Abbreviations

ABB	Asea Brown Boveri; one of the biggest electrical- and automation technology industry corporations
AC500-eCo	PLC product family at ABB
ACH550	Product family of frequency converters at ABB
ACS550	Product family of frequency converters at ABB
ACS600	Product family of frequency converters at ABB
ACS800	Product family of frequency converters at ABB
ACS880	Product family of frequency converters at ABB
ASCII	American Standard Code for Information Interchange
CIA	Central Intelligence Agency; a civilian foreign intelligence service of the United States Government
CLU	Choke Load Unit; a substitute for a motor when loading the frequency converter
CPU	Central Processing Unit; the electronic circuitry within a computer that carries out the instructions of a computer program
CRC	Cyclic Redundancy Check; a type of hash function used to produce a checksum in order to detect errors in data storage or transmission
DSW	Drives Service Workshop; 21 ABB repair shops around the globe, from which one is located at Vantaa

FTP	File Transfer Protocol; a standard network protocol used to transfer computer files between a client and server on a computer network
IGBT	Insulated Gate Bipolar Transistor
I/O	Input / Output
NETA-21	ABB’s Remote Monitoring Tool
NSA	National Security Agency; an intelligence organization of the United States Government
PID	A Proportional–Integral–Derivative controller is a control loop feedback mechanism commonly used in industrial control systems
PLC	Programmed Logic Controller; a digital process controller
PWM	Pulse-Width Modulation; a technique for encoding information into a signal like sine wave
RMBA-01	ACS800 Modbus connection device
RTC	Real-Time Clock; the clock that keeps the current time while a computer is turned off
RTU	Remote Terminal Unit; a microprocessor controlled electronic device
TCP/IP	Transmission Control Protocol / Internet Protocol; the computer networking model and set of communications protocols used on the Internet and similar computer networks

1 Introduction

Because knowledge is power and power is knowledge, nowadays data is gathered on servers everywhere, so it could be later used. Good examples of this are giant corporations like Google and Microsoft or international organizations like NSA and CIA. Google has gathered data on its serves for years from all over the globe, from its users and what is searched in the search engine. A Google data centre is also found in Finland, in Hamina. Microsoft entered this same doing in launching the Windows 10 operating system, which gathers almost any information possible on its users. There is also the widely known case of Edward Snowden, who revealed several top secret data of NSA and CIA in 2013, on how to gather knowledge from people. [1; 2]

This project on the other hand is not about spying for information, although it is about data and knowledge. The idea of this project came from a customer's need of getting proof of how and with what kind of ranks their frequency converter has passed the electronic tests in ABB's DSW repair shop. Before starting this project, it was not possible to get any data out of most of the test environments.

In Vantaa's DSW repair shop there are at least three test areas, which the method could be used in. One of these test areas is equipped with a choke load and a few small motors. The motors are used only for frequency converter's identification. There are choke loads to be installed to the other test area during 2016 summer. The big motors already there will be backups and for ID-drives. In the third test area there are multiple different sized motors, which can be used for loading and ID-drive.

The objective is to execute a system with programmable logic, which would collect parameter information from the frequency converter during testing and save the information to its own memory in charts, where it would be collected from with a computer, using for example a FTP connection. The files should be able to be opened directly with a normal chart program, like Microsoft Excel. The first layout plan of the project is seen in appendix 1.

2 ABB Ltd. and DSW Repair Shop

2.1 ABB Ltd.

ABB or Asea Brown Boveri is one of the biggest electrical- and automation technology industry corporations. It produces everyday machinery for electrical engineering professionals and ordinary consumers according needs. The headquarters is based in Zürich, Switzerland. ABB functions in over 100 countries and employs over 100 000 people. In Finland ABB has expanded throughout the country, but the biggest campuses are in Helsinki and Vaasa. There are already over 7000 employees in Finland and the turnover exceeds 1.7 billion euros. The frequency converter factory in Pitäjänmäki, Helsinki is the biggest in the world. [3]

2.2 Drive Service Workshop Finland

ABB Drives Service Workshop Finland is one the main DSWs in ABB. It has a special role in the DSW network because of long history drives manufacturing in Finland where main responsibility of AC Drives still is. Closer cooperation with Product Engineering's Reliability and Quality functions means that DSW Finland often is the one which ramp-up workshop repair services for new product families. There are many frequency converter product families, but at the moment the most engaging ones in the Vantaa repair shop are ACS550, ACH550, ACS600 and ACS800. ACS880 is just on the way to the repair shops, because it represents the new generation of ABB. DSW Finland also ramp-down repairs some older products before those are going to obsolete phase. The oldest equipment may have been manufactured already in the 1970's. Because of close cooperation with Product Engineering, DSW Finland also from time to time takes care of the most challenging cases and analyses.

2.3 Electrical Measurements in Repair Work

When a frequency converter arrives to the repair shop to get repaired, first of all it is checked visually and it is stated if it is repairable. If it is repairable, a couple of measurements with a multimeter can be made.

At least these sections are always measured;

- The main circuit, to see if the thyristors or IGBTs are in short circuit
- Capacitors

- Discharge- and Charging Resistors
- Fuses

In the bigger frame sizes R7-R9 different components are measured in a little more detail with the multimeter, and a test is made to the AGDR card with ABB’s specified tester, so it is seen if IGBT/AGDR-unit is faulty.

If there are no faults found in the frequency converter, the next step is to make an insulation resistance measurement to it. The insulation resistance measurement tells if there is danger of a short circuit and if the insulators are unbroken.

After these the drive test is set in full voltage, and start in scalar control mode. If this also works without any failures, the device is connected to the test line, and an ID run is performed. If the ID run also pass, the device is loaded in the test line for 1-2 hours. If after all these tests the frequency converter does not have any fault with it, it is stated as intact. However, if a fault is found in the frequency converter, it will be repaired and the tests are performed again after the repairing from the scalar run onward. In figure 1 is seen the ACS800-104 frequency converter in the insulation resistance measurement and the equipment with which a full DC-voltage into the frequency converter can be added.

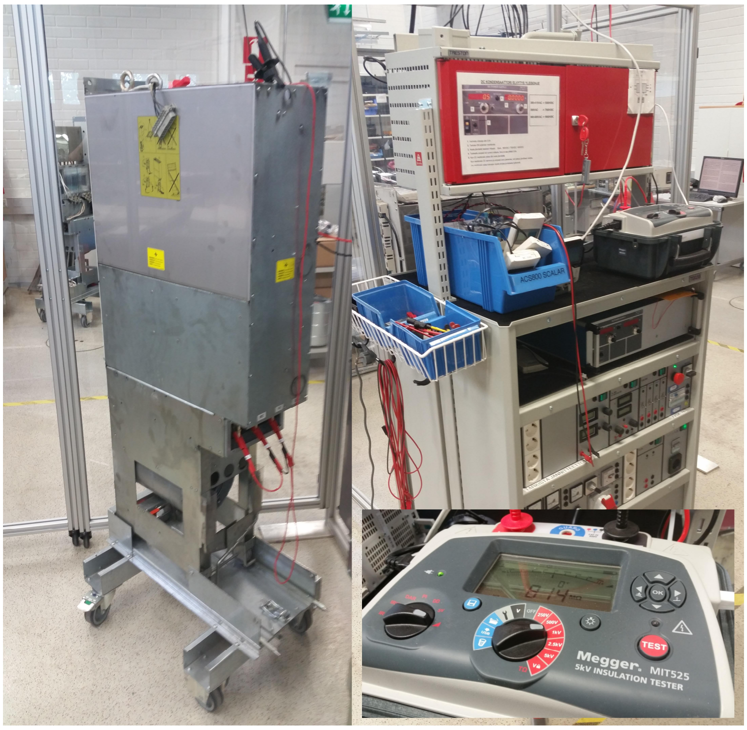


Figure 1. ACS800-104 in the insulation resistance measurement.

3 Programmed Logic Controller (PLC)

3.1 PLC in General

For decades programmed logic controllers (PLCs) have been a part of factory automation as well as industrial process control. ABB has many own PLC products, and this project is about solving the possibility of using them. PLC allows to control the frequency converter and a distant data collection from it.

In practice, programmable logic controller (PLC) is a small computer equipped with a microprocessor, which is used in controlling the real-time automatic processes. Originally it was harnessed in the car industry, where software updates replaced the rewiring of the control systems. There it spread to other industries, because one can easily replace hundreds or thousands of previously used relays and timers with one logic. The programmable logic controllers (PLC) functionality has slowly grown from a traditional relay substitute to a programming center which manages the developed movement control, process adjustment, diversified control systems and networks. [4]

In one programmable logic controller (PLC) there can be several modular and integrated input and output gates, which have been attached with sensors and actuators in the field. The logic can guide the actuators with user created programs and data given by the sensors. The program has been located in a battery secured memory. Through the input gates the logic gets information on the state of the system, and through the output gates it can control the system. All digital signals act like switches, they only indicate on- and off –statuses (1 or 0, true or false). To identify digital signals, usually voltage or current is used. In this case the certain space of the quantity is interpreted as 0 –space and the other one as 1 –space. [4]

Originally in the programmed logics there were only digital interfaces, but nowadays they are attached with analogic connections. The analogic signals act like volume controllers. They transmit all values between the farthest ends of its area of operation. Usually these margins are interpreted as integers. The actual accuracy is dependent of the used equipment and better accuracy acquires more memory. In most cases the measurement data transmitted through the analogic signal are pressure-, current-, and temperature values. The value of the measurable signal can be anything in the chosen measurement area. [4]

The dispersion of the I/O in the field as well as connecting intellectual actuators and sensors in logic in enabled by fieldbus technology. In these cases the quantities of transferred data can be much larger than by traditional wired technology. The transferred data can also already be refined by the own intellectuality of the field device. Fieldbus technologies are for example Profibus DP and Modbus. More information on the latter will be found in chapter 5. [4]

The programs for the logics are usually written by computer with a program especially intended for it, and afterwards transferred to be executed in the battery secured RAM –memory or other permanent memory. Most of the time the programs are subjective according to the manufacturer. The producers of the devices usually offer completed collection of programs to their own logics, at which time different functions such as PID –adjustment or RTC –clock are easy to execute with finished program sectors. Using these completed sectors eases the programming and adds the reliability of operation, because someone has already tested them and proved to be functional. [4]

Because the programmable logic controller (PLC) performs the assignments just as specified in the program, the operator of the program must take in count all the different circumstances in advance when composing it. Solving of abnormal situations and making process modifications can be done with a user interface. The interface can be a PC and a control room, or an operating panel through which the controller of the process gets real time data of its situation in charts, alarms, position based displays or reports. [4]

3.2 PLC in ABB

ABB has two product families in the programmable logic market, AC500 and AC800. AC800 contains remarkably more expensive and more efficient models, so that was not chosen for this project, as there was no need for those features. [5]

The AC500 product family contains four different sub product families for small, middle and high-end applications, which are;

- AC500, powerful flagship PLC with a wide range of performance, communications and I/O capabilities for industrial applications

- AC500-eCo, offers flexible and economical configurations for your modern control system
- AC500-S, for safety applications involved in factory or machinery automation area
- AC500-XC, for extreme conditions with extended operating temperature, immunity to vibration and hazardous gases

The AC500-eCo used in this work is based on the basic model AC500, but costs less and has trimmed down on some of the features. However, it has all the necessary features for a modern control system in small applications. CPU comes with one RS-485 serial interface and the other can be added as a second option. CPU can be expanded by up to ten I/O modules, which means up to 335 local I/O channels. If a CPU with Ethernet capabilities is selected, it has a Web server, FTP server and SMTP client. With a Web server, the received PLC's results can be entered directly to any Internet site which you have rights to. With the FTP server, a connection between the PLC and a PC can be made, which allows to easily import PLC data to the computer. SMTP client is able to send the collected data directly to an email. In figure 2 ACS500-eCo PLC's CPU is seen. [5]

Figure 2. AC500-eCo PLC's CPU. [5]

4 Frequency Converter

4.1 Frequency Converter in General

A frequency converter is mainly used in controlling the speed of a three phased induction motor, but it also has several other functions. The size ranges of frequency converters vary starting from 0.1kW to over 10MW devices. In process industry the most common input voltage for frequency converters varies between 400-690V of three phased alternating current. The input voltage does not necessarily need to be three phased; for example in small powered frequency converters can be input with 230V AC one phased network, at which time the frequency converter forms the motor a 3x230V adjustable three phased shift voltage. In such cases the coil voltage of the engine must be 230V, the same as the phase voltage. In some special cases the frequency converter can be built up to a 6kV high voltage. [6, p. 136]

With a frequency converter a three phased alternative current is created. Its starting frequency can be adjusted, regardless of the grid that feed it. Most of the time in normal use, the frequency is between 0-50Hz. In some special cases it can be raised up to 7000Hz. The engines stamina does not have to be jeopardized, because while the motor is started, sufficient torque by adjusting voltage and current is made. [6, p. 137]

In those devices where the counter torque curve is quadratic, the biggest energy savings are made. This means that the load will excavate in the square of the spinning speed and the consumption in the cube of the spinning speed. These usage targets are for example blowers and pumps. [6, p. 137]

The most common and most simple frequency converter type is a three phased voltage source inverter. It contains a rectifier, filter- and capacitance circuit, which works as an intermediate landing of electrical energy, and inverters. First the input frequency is rectificated into a direct current with a diode- or thyristor bridge. After this the electrical power with a combination of a choke and capacitor is filtered. The next phase is a switch faction. There switch components are used, which are formed by the number of outlet phases of thyristor- or IGBT transistor pairs. One component of each pair will conduct at a time with rapid phase, using for example pulse width modulation. The originated alternating current is formed as a variable squad wave, but before output the filtered voltage is near the sine wave. In the next page in figure 3 a circuit diagram of a PWM frequency converter is seen. [7]

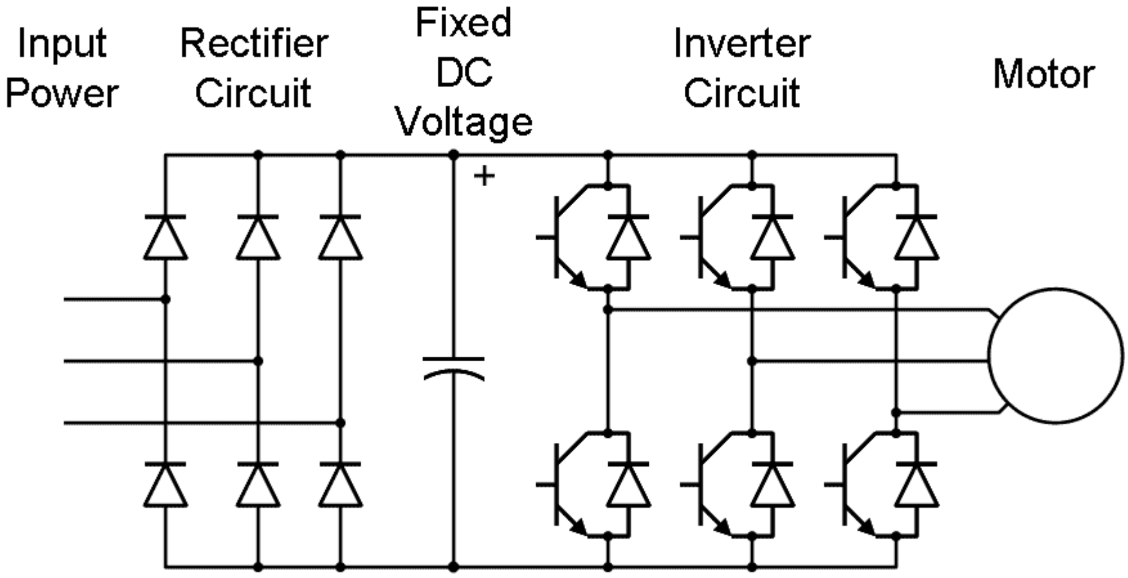


Figure 3. PWM Frequency converter´s diagram. [7]

4.2 Frequency Converter in DSW Finland

The ACS800 is at least at the DSW Finland the most repaired and maintained frequency converter. It can be found in a number of different types, different frame sizes and with different cooling systems. The wall-mounted frame sizes are R2-R6. They are always air-cooled and can be found in all sizes with IP21 and IP55 ratings. These frame sizes can be found devices for either single- or multi drive and in some cases these are also installed in the cabinet. The frame sizes R3-R6 are shown in figure 4.

The following frame sizes R7-R9 are mainly multi drives, but frame sizes R7 and R9 are available as single drive too. They are also most commonly air-cooled, but of these frame sizes at least R8 can also be found in water -cooled versions. These larger frame sizes are always installed in a cabinet. The devices can be obtained with a number of different power classes and voltage classes, as well as a variety of different options. Shown in figure 5 there are cabinet lines which these devices are installed on and two pcs of air-cooled ACS800-104 R8 inverters, as well as their ALCL filter unit.

Figure 4. ACS800 frame sizes R3-R6. [3]

Figure 5. Two pcs ACS800-104 R8 air cooled frequency converters and one ALCL filter unit.

5 Modbus

Modicon published a serial communications protocol in 1979, which was named Modbus. This protocol was meant to be used with Modicons programmable logics (PLC). However, this protocol was formed into a standard in the industry, and is now widely used in the communication of electronic devices. Mainly this is because Modbus is open and free of license toll, an easily deployed industrial network, which transfers raw data without the restrictions placed by device manufacturers. It enables the communication between several devices linked to the same network. The protocol has serial port- and Ethernet versions and it is used to connect the monitoring computer with a field device (RTU) in centralized management systems. [8; 9]

There are two modifications to serial formed traffic with different numeric data forms and protocol details. These formations are named Modbus/RTU and Modbus/ASCII. RTU is a compact binary data form and ASCII is a text based and understandable by humans. In RTU, CRC –checksum is used for integrality control, whereas longitudinal redundancy check – checksum is used in ASCII. These protocols may not be used mixed because the knots that are used to define to use an RTU protocol, are not communicating with knots that use ASCII protocol, and vice versa. [8; 9]

A newer protocol has been developed for TCP/IP connections, called Modbus/TCP. This protocol is easier to implement than Modbus/RTU or Modbus/ASCII, because calculating of the checksum is not needed. Data model and function calls are similar for all three connection protocols, only the encapsulation is different. [8; 9]

For each device installed to a Modbus lane a unique address is defined. Each device can send a Modbus command, but usually only one master device will do it. The command includes that device's address, which the command is meant for. Only this device will perform this command, although all devices can receive commands. A command also always includes a checksum, which confirms the correct passage of the command. The basic commands can order a field device to switch some registers value, or order the device to send back one or more values of its registers. [8; 9]

Because Modbus was designed in the late 1970's to combine programmable logics (PLC), its data type quantities are limited to those that were then used, and so big binary objects are not supported. To field equipment there is no standardized way to find a description of a data object, like to define that a register value means temperature between 15 and 100. With field equipment there is no possibility to report exceptional situations, because Modbus is a host/slave protocol. A host device must ask every field equipment information regularly, which is slow in most devices applications and takes a lot of capacity from the network, for example in slow radio links. The address of the protocol has been restricted to 254 devices in one lane. Because the data transfer must be continuous, it limits the distant connections to such that can buffer data to avoid disconnections. [8; 9]

6 NETA-21

NETA-21 is a distant monitoring module for frequency converters designed by ABB. NETA-21 has a built-in web server, which works with most known web browsers. Distant monitoring is possible on all browser based products, like computers, tablets and smartphones. With the module, monitoring and maintaining unmanned applications distantly is possible. This possibility can be used in many different industry fields which have long distances between maintenance destinations, such as construction industry, oil- and gas industry and eater- and wind power plants. This intelligent tool has all needed programmes for the user interface, data transmission and conservation. NETA-21 can also be used as a distant PC substituent, so a local computer would not necessarily be needed at the user destination. [10]

In addition to distant monitoring, with NETA-21 it is able to read and save information on passed events and real time occurrences. The module allows reading and adjusting frequency converters parameters and studying statuses. It also has a defect log. It is possible to save the data in the defect log and for example send into the users email. As a memory location NETA-21 uses an SD memory card, where attracting information and saving it into a FTP-server is possible. The reports can be sent straight to an email between certain time periods or as real time information. NETA-21 can be programmed to report automatically when a defect or a warning is detected. By default the module saves all changed usage happenings and parameters on an SD card daily. On top of that it is possible to create your own reporting format and influence the email reporting yourself. [10]

NETA-21 can be joined with the frequency converter with a few different ways; either a RJ45 –connection directly to the frequency converters panel port or via FENA-11 module with an Ethernet –cable. There are 2 RJ-45 ports in NETA-21, and up to 32 applications to one port can be attached. Using the panel ports an own panel to the frequency converter cannot be attached, because NETA-21 is fastened in the panel port of the frequency converter. Hence, programming straight from the frequency converters own monitor is not possible. [10]

When wanting to monitor the usage, the IP address of the NETA-21 module to the browser has to be written down. This allows the program to open a view of the devices. To get in touch with the devices, the program requires a user account and password, which prevents outside access to the drive. It is possible to make different views to particular users and limit their authority to the program. The software also enables monitoring operations from several places at the same time. [10]

By a browser based program controlling the operations will not work, for its function is only to monitor usage. With the actual program reading and setting parametric values and studying status data and its real time values from usage is possible. If a fault has already passed, it can be checked out with the program. The software also has a data logger and an error log. The info from both to an own file in a FTP server or an SD card can be saved. ABB's product NETA-21 is seen in figure 6. [10]

Figure 6. ABB's product NETA-21. [10]

7 Execution

7.1 “Kuristamo” and ABB’s Choke Load Unit (CLU)

During the project the aim was to install the finished data collection system to ABB’s Vantaa DSW repair shops test area called “Kuristamo”, where CLUs, choke loads, are used instead of motors in the frequency converters load run. In “Kuristamo” it is possible to run with four different choke. The choke sizes are 25A, 38A, 61A and 87A. Unfortunately it did not get to a point in this project, where the PLC could have been installed to the test line. This was because the programming in the PLC was not in the state where it could save the acquired data. In figure 7 an overview of test area “Kuristamo” is visible.

Figure 7. ABB DSW Finland test area ”Kuristamo”.

7.2 NETA-21 vs PLCs

Because both devices would be suitable for this project, it needs to be explained why PLC was chosen to be used. It was clear as soon the project started, that PLC would be used instead of NETA-21. NETA-21 is a little outdated technology at ABB and its time is soon over. That is why a new generation AC500-eCo PLC was chosen, which has broad possibilities to expand, but is still reasonably priced compared to other AC500 series PLC. PLC can also be used to control a choke load, if it would be wanted to make the testing fully automated. The PLC used in the project with extra accessories costs about 400-500 euros. The processor unit (CPU) that was chosen is PM556-TP-ETH model, with additionally installed TA562-RS-RTC and MC503 SD-card adapter.

7.3 Saving Parameters from ACS800 with AC500-eCo PLC

In the beginning of the project the idea was that it could be possible to take connection via PLC to ABB's ACS800 frequency converters straight through a panel port using a Modbus connection and RJ-45 cable. After multiple tests it was discovered that this was not possible. That is why an RMBA-01 adapter module option was chosen to be used, which is designed for ACS800 frequency converters. A RMBA-01 option is easy to install. It is attached to the frequency converters option slot 1 with two screws. In the deployment it is checked that the frequency converters parameters part 52 settings are alright. In this project, baud rate 9600bit/s, parity odd, data bits 8 and stop bits 1 were chosen as the connection settings. The same settings should be used in PLC program when defining the Modbus settings. In figure 8 are the Modbus parameters of PLC COM2, specified by Automation Builder. In figure 9 the PLC Modbus server settings are seen. [11]

Parameter	Type	Value	Default Value	Unit
Enable login	Enumeration of BYTE	Disabled	Disabled	
RTS control	Enumeration of BYTE	Telegram	None	
Telegram ending value	WORD(0..65535)	3	3	
Baudrate	Enumeration of DWORD	9600	19200	Bits/s
Parity	Enumeration of BYTE	odd	even	
Data Bits	Enumeration of BYTE	8	8	Bits/character
Stop Bits	Enumeration of BYTE	1	1	
Run on config fault	Enumeration of BYTE	No	No	
Operation mode	Enumeration of BYTE	Client	None	
Address	BYTE(0..255)	0	0	

Figure 8. COM2 Modbus Parameters.

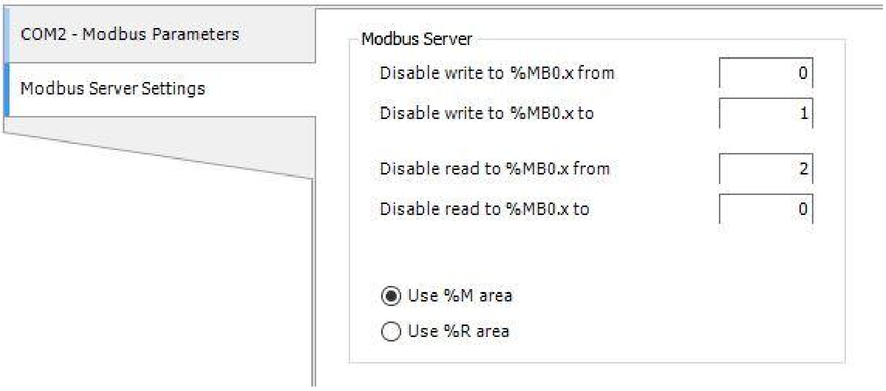


Figure 9. Modbus Server Settings.

In order for the PLC to do anything, it must be programmed. For making the program, ABB's own computer program called Automation Builder was used. Automation Builder uses Codesys as its programming environment, which is an international programming environment based on IEC 61131-3 standard. In Codesys all five programming ways according to the IEC 61131-3 standard are found, and one which does not fit to the IEC standard. The programming manners can be divided into two groups, which are text- and graphical editors. [4; 12; 13; 14]

Text editors:
Instruction List (IL)
Structured Text (ST)

Graphical editors:
Ladder Logic (LD)
Function Block Diagram (FBD)
Sequential Function Chart (SFC)
Continuous Function Chart (CFC)

The first sequence starts once the PLC is connected to the RMBA-01 option based in the frequency converter and both devices are turned on. The PLC program first reads six parameters from the parameter group 99 of the frequency converter starting from 99.03, and saves the parameter values to chart "Arvot_99". After this the program grows the sequence by one and the next function gets started.

In the next activity, the second sequence, the PLC program reads nine parameters from the parameter group 01 starting from 01.01 and saves the parameter values to chart “Arvot_01”. There the values are one at a time transferred to “Muisti[Array_laskuri.CV]”. The program repeats this reading- and saving process as many times as it is defined in the programming, whereas the “Array_laskuri.CV” defines which memory spot/process is in question. An example program repeats the cycle fifty times every second, but both values are adjustable by the program. When the program has repeated the process a definite amount of times, the program again increases the sequence by one and the next function is started. In figure 10 it is seen how the PLC program has read the desired values of the frequency converters parameter data and collected them to variables Arvot_99 and Arvot_01.

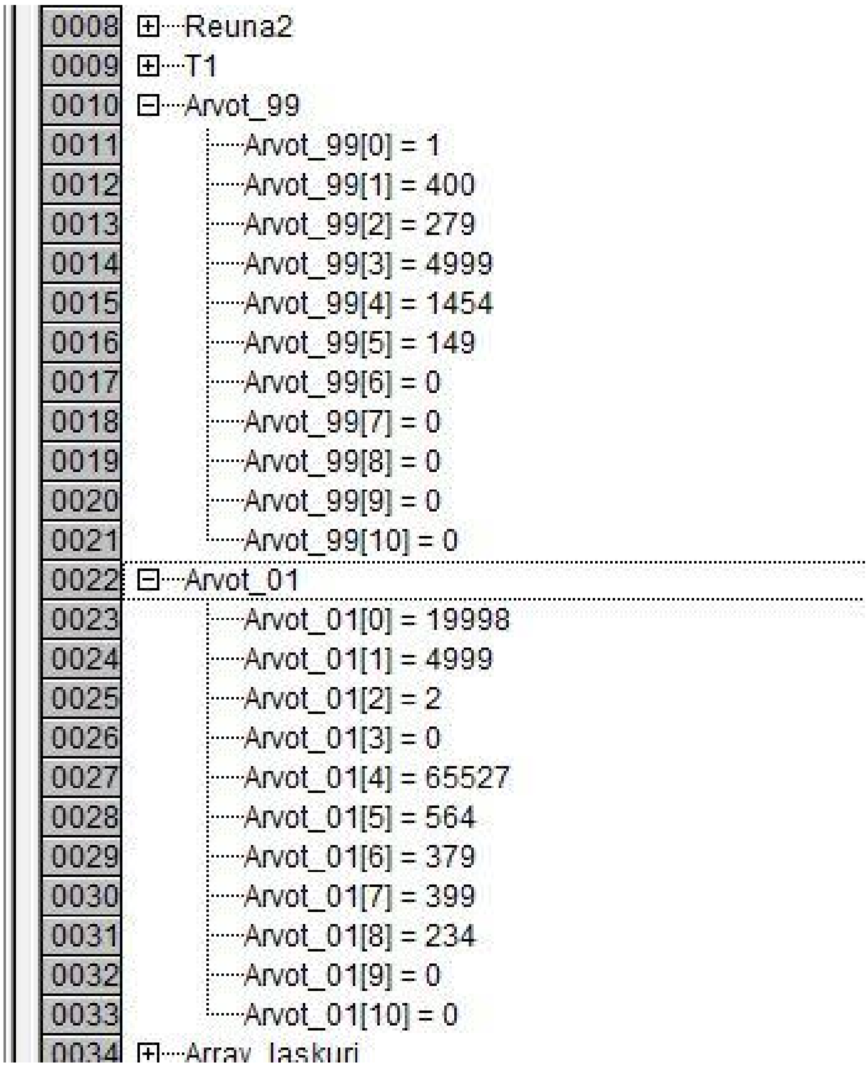


Figure 10. Collected parameters from frequency converter to Arvot_99 and Arvot_01.

In the third sequence the program is meant to opens a text file in the PLC's memory file. In the text file the program first types the headlines, separated with a semicolon, and afterwards types from the chart ”Muisti” the parameter values in the next line also separated with a semicolon, switching the row each time the ”Muisti” line changes. Unfortunately this part does not function correctly in the program that was made and it would have to be modified. When all the writing is done, the program closes the file and again increases the sequence by one. In the next page in figure 11 is seen how the parametric values are saved in the charts “Muisti”.

- Muisti
  - Muisti[0]
    - .Nopeus = 0
    - .Taajuus = 0
    - .Virta = 0
    - .Momentti = 0
    - .Teho = 0
    - .DC_Jannite = 0
    - .Lampotila = 0
  - Muisti[1]
    - .Nopeus = 19999
    - .Taajuus = 4999
    - .Virta = 2
    - .Momentti = 0
    - .Teho = 65527
    - .DC_Jannite = 563
    - .Lampotila = 229
  - Muisti[2]
    - .Nopeus = 19998
    - .Taajuus = 4999
    - .Virta = 2
    - .Momentti = 3
    - .Teho = 65527
    - .DC_Jannite = 563
    - .Lampotila = 229
  - Muisti[3]
    - .Nopeus = 19998
    - .Taajuus = 4999
    - .Virta = 2
    - .Momentti = 1
    - .Teho = 65527
    - .DC_Jannite = 563
    - .Lampotila = 229
  - Muisti[4]
    - .Nopeus = 19998
    - .Taajuus = 4999
    - .Virta = 2
    - .Momentti = 0
    - .Teho = 65527
    - .DC_Jannite = 564
    - .Lampotila = 229

Figure 11. An example of the parametric values in the charts ”Muisti”.

The PLC saves the chart formed from parameter values in text form to an inner user disk, where the file can be found via FTP connection. The ways to form a FTP connection between PLC and a computer are multiple. The two ways that were tried are either a third party free application called FileZilla or cmd.exe found in Windows, called Command Prompt. In figure 12 there are the PLC FTP server settings with Automation Builder.

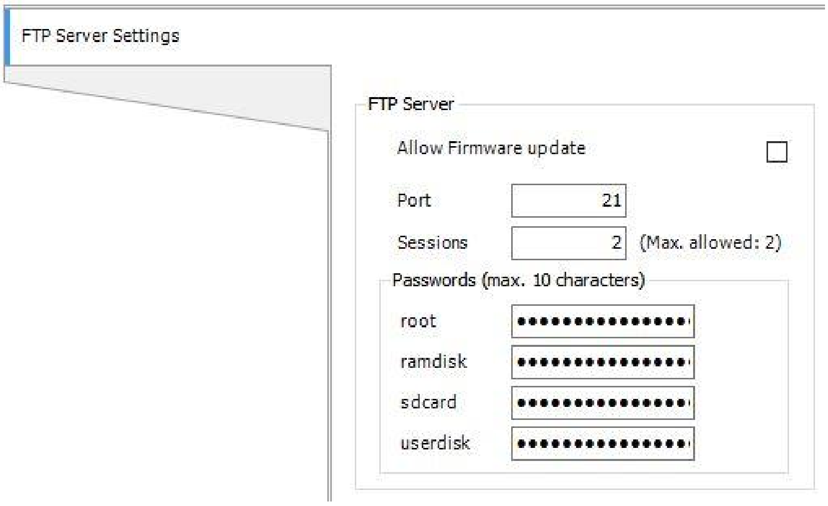


Figure 12. FTP Server Settings.

In order to retrieve the file from PLC user disk with a Command Prompt, one must write these commands in correct order;

/ftp	
/open *IP-address*	f.ex. /open 192.168.0.10.
Password	f.ex. userdisk
Password again	f.ex. userdisk
/dir	
/get *file name*	f.ex. /get testi.txt

8 Results

In order to get the project to a stage where it can be used in the test lines, the design of the PLC program should be finished. At this moment the PLC can proficiently read the parameter values of the frequency converter and save that information to charts in the PLC's memory. In appendix 2 the program I made for the PLC can be found. However, when it is tried to save the parameter values in a text file in a rational way, there are some errors and the wanted result will not be shown. In the figures 13 and 14 below are the models of what the results should look like in text files and Excel –files. At this moment only vague text or a first text row is saved in to a text file.

```
Aika;DC_Jannite;Lampotila;Momentti;Nopeus;Virta;Taajuus;Teho;
11:05:13;0;0;0;0;0;0;0;
11:05:53;563;22;85;99;45;49;899;
11:06:33;563;27;86;99;50;49;905;
11:07:13;562;33;86;99;48;49;911;
11:07:53;564;40;85;99;49;50;908;
11:08:33;563;48;86;99;46;49;901;
```

Figure 13. Example texts file from collected data.

Aika	DC_Jannite	Lampotila	Momentti	Nopeus	Virta	Taajuus	Teho
11:05:13	0	0	0	0	0	0	0
11:05:53	563	22	85	99	45	49	899
11:06:33	563	27	86	99	50	49	905
11:07:13	562	33	86	99	48	49	911
11:07:53	564	40	85	99	49	50	908
11:08:33	563	48	86	99	46	49	901

Figure 14. An example Excel chart from texts file.

In addition regarding the future, the PLC program should be modified to match the ACS880. FSCA-01 option can be used as an ACS880 Modbus connection device, which is a RS-485 adapter module designed for ACS880.

9 Conclusion

The end results were not exactly what was hoped for in the beginning of the project. Making a data collecting tool from the beginning requires a lot of programming knowledge and development. The result of this project, however, is very usable for ABB, when it is further developed. I myself learned a lot of programmable logics and the operations during this time, since before I had no experience in that field. The best results might have been achieved if someone else took over the PLC programming, for example professionals inside ABB or an outside party, and the designing and reporting of the work would have been defined for one person.

This project had a really good and practical idea, and there lies several possibilities in utilizing it further. These were the first steps in making the data more available from the test systems. The recommended follow-up is to have someone who knows about PLC programming to finish the program, so that writing in a text file would also be possible. From there, the collected values could easily be transferred to Excel where drawing graphics is easy.

References

1 Wikipedia. Edward Snowden. URL: https://fi.wikipedia.org/wiki/Edward_Snowden. Accessed 13 October 2015.

2 Google Data Centers. Hamina. URL: https://www.google.com/about/datacenters/inside/locations/hamina/. Accessed 8 February 2016.

3 ABB. Official Website. URL: http://www.abb.com/. Accessed 9 May 2016.

4 Wikipedia. Programmable logic. URL: https://fi.wikipedia.org/wiki/Ohjelmoitava_logiikka. Accessed 14 April 2016.

5 ABB. PLC. URL: http://new.abb.com/plc. Accessed 10 May 2016.

6 Markku J.J. Mäkinen, Raimo Kallio. Teollisuuden sähköasennukset. Helsinki: Otava; 2004.

7 Wikipedia, Frequency converter. URL: https://fi.wikipedia.org/wiki/Taajuusmuuttaja. Accessed 20 April 2016.

8 Wikipedia. Modbus. URL: https://fi.wikipedia.org/wiki/Modbus. Accessed 12 April 2016.

9 Modbus. Official Website. URL: https:// http://www.modbus.org/. Accessed 16 May 2016.

10 ABB Library. NETA-21 user´s manual. URL: https://library.e.abb.com/public/5ad2a0e9056a171ec1257dc800390dfa/NETA_21-UM_revD_screen_A4.pdf. Accessed 29 September 2015.

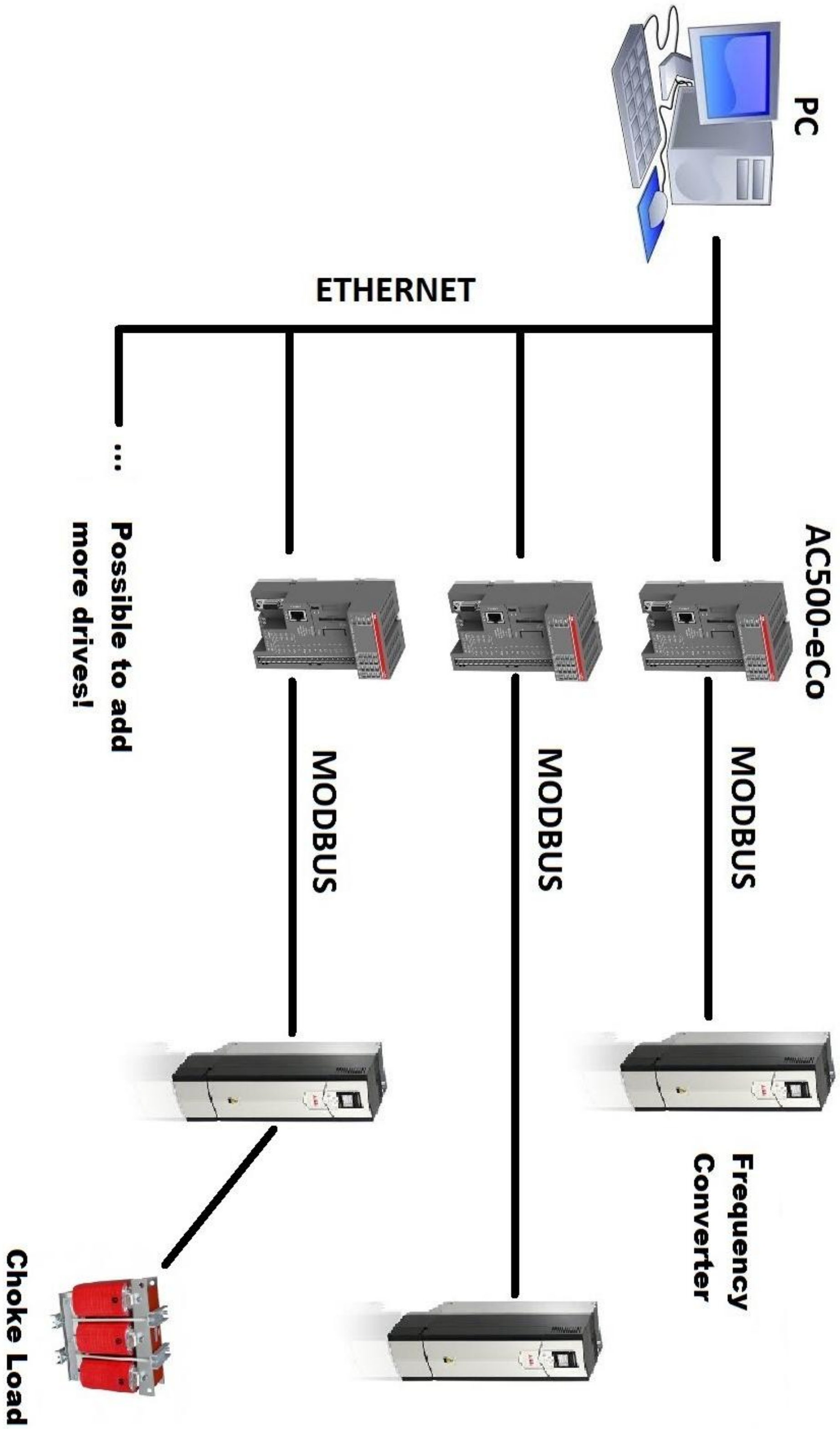
11 ABB Library. RMBA-01 user´s manual. URL: https://library.e.abb.com/public/684b1e6f7e657ef5c1256d2800400664/EN_RMBA_UM_A.pdf. Accessed 19 April 2016.

12 Wikipedia. Codesys. URL: https://fi.wikipedia.org/wiki/Codesys. Accessed 30 March 2016.

13 Wikipedia. IEC 61131-3 –standard. URL: https://en.wikipedia.org/wiki/IEC_61131-3. Accessed 12 April 2016.

14 ISA. IEC 61131-3 –standard. URL: https://www.isa.org/standards-publications/isa-publications/intech-magazine/2012/october/system-integration-iec-61131-3-industrial-control-programming-standard-advancements/. Accessed 16 May 2016.

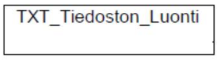
Layout


PC
ETHERNET
AC500-eCo
MODBUS
MODBUS
MODBUS
...
Possible to add more drives!
Frequency Converter
Choke Load

Program

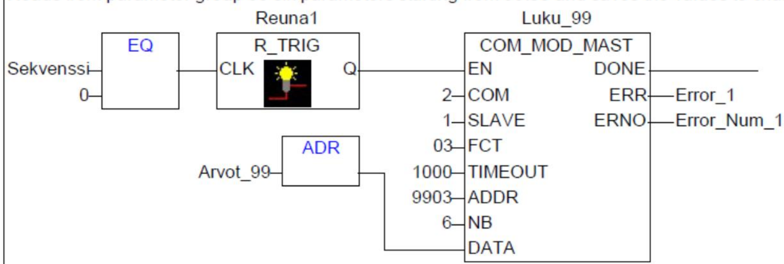
```
PROGRAM PLC_PRG
VAR
    Luku_99: COM_MOD_MAST;
    Reuna1: R_TRIG;
    Eka_lohko: WORD;
    Toka_lohko: WORD;
    Luku_01: COM_MOD_MAST;
    Reuna2: R_TRIG;
    T1: BLINK;
    Arvot_99: ARRAY [0..10] OF WORD;
    Arvot_01: ARRAY [0..10] OF WORD;
    Array_laskuri: CTU;
    Error_1: BOOL;
    Error_Num_1: WORD;
    Error_2: BOOL;
    Error_Num_2: WORD;
    X1: WORD;
    X2: WORD;
    Timer1: TON;
    Aika: TIME;
    Laskurin_reset: SR;
END_VAR
```

0001



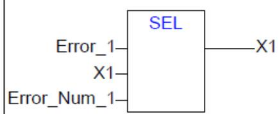
0002

Reads from parameter group 99 six parameters starting from 99.03 and saves the values to chart Arvot_99



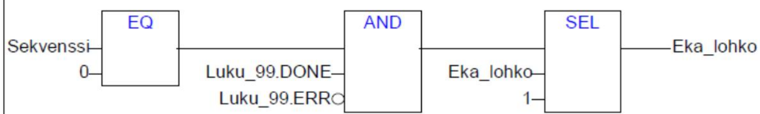
0003

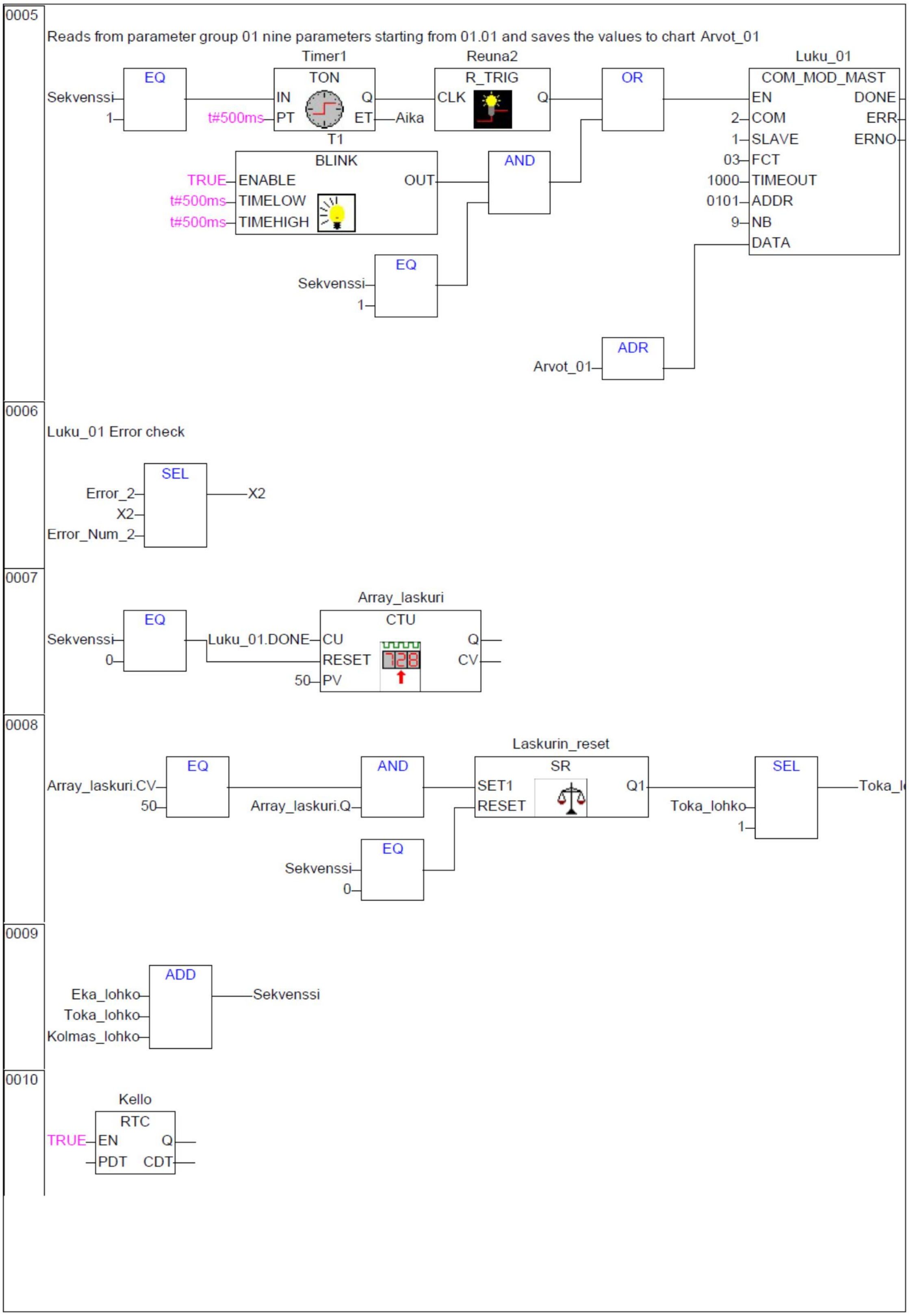
Luku_99 Error check



0004

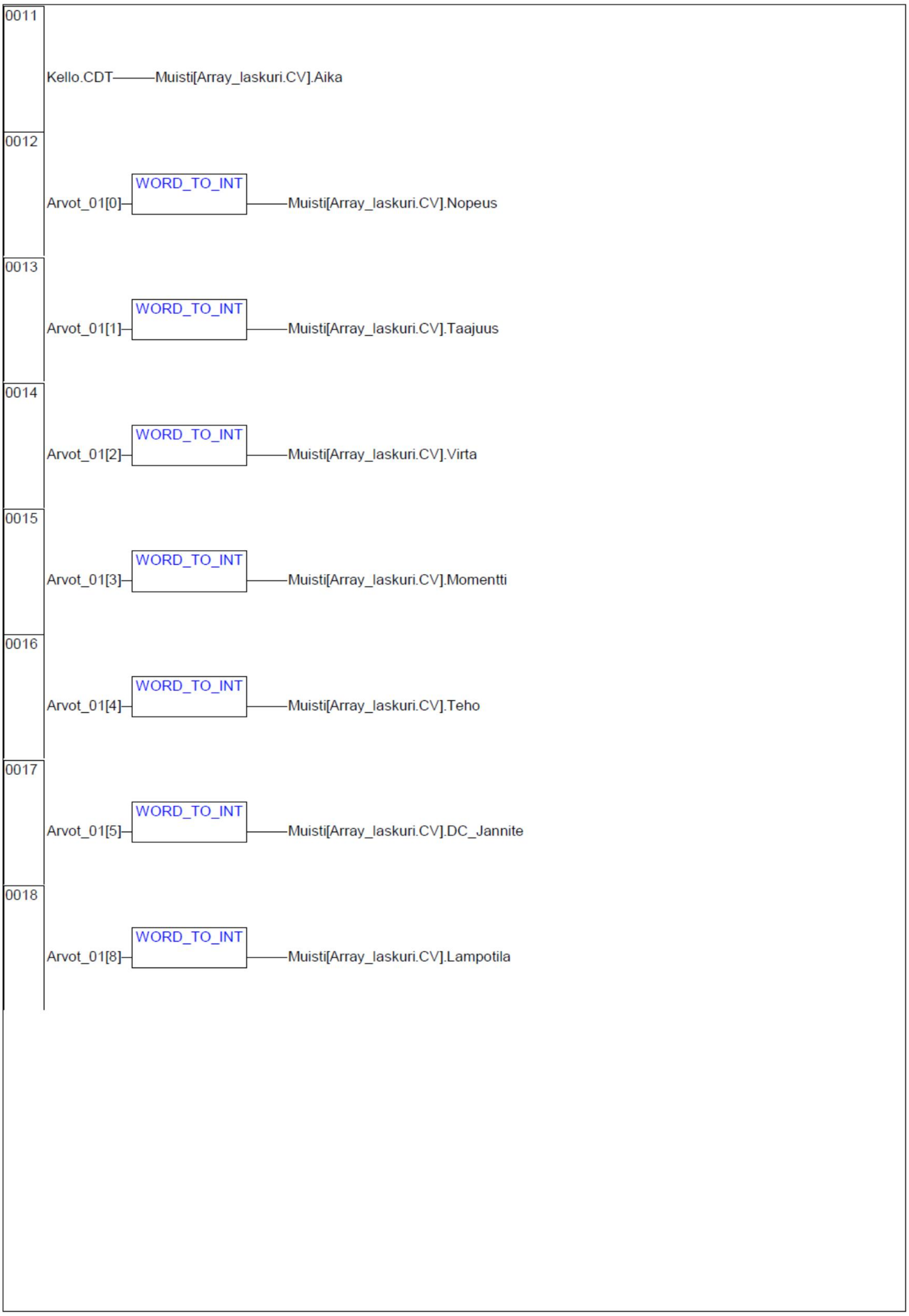
When Luku_99 is ready, grows sequence by one




0005
Reads from parameter group 01 nine parameters starting from 01.01 and saves the values to chart Arvot_01
Timer1
Reuna2
Luku_01
EQ
Sekvenssi
1
TON
IN
PT
Q
ET
Aika
t#500ms
R_TRIG
CLK
Q
OR
COM_MOD_MAST
EN
COM
SLAVE
FCT
TIMEOUT
ADDR
NB
DATA
DONE
ERR
ERNO
2
1
03
1000
0101
9
T1
BLINK
TRUE
t#500ms
t#500ms
ENABLE
TIMELOW
TIMEHIGH
OUT
AND
EQ
Sekvenssi
1
ADR
Arvot_01
0006
Luku_01 Error check
SEL
Error_2
X2
Error_Num_2
X2
0007
Array_laskuri
EQ
Sekvenssi
0
Luku_01.DONE
CTU
CU
RESET
PV
50
Q
CV
0008
Laskurin_reset
EQ
Array_laskuri.CV
50
AND
Array_laskuri.Q
SR
SET1
RESET
Q1
SEL
Toka_lohko
1
EQ
Sekvenssi
0
0009
ADD
Eka_lohko
Toka_lohko
Kolmas_lohko
Sekvenssi
0010
Kello
RTC
TRUE
EN
PDT
Q
CDT

Error_2
Error_Num_2

ohko


0011
Kello.CDT
Muisti[Array_laskuri.CV].Aika
0012
Arvot_01[0]
WORD_TO_INT
Muisti[Array_laskuri.CV].Nopeus
0013
Arvot_01[1]
WORD_TO_INT
Muisti[Array_laskuri.CV].Taajuus
0014
Arvot_01[2]
WORD_TO_INT
Muisti[Array_laskuri.CV].Virta
0015
Arvot_01[3]
WORD_TO_INT
Muisti[Array_laskuri.CV].Momentti
0016
Arvot_01[4]
WORD_TO_INT
Muisti[Array_laskuri.CV].Teho
0017
Arvot_01[5]
WORD_TO_INT
Muisti[Array_laskuri.CV].DC_Jannite
0018
Arvot_01[8]
WORD_TO_INT
Muisti[Array_laskuri.CV].Lampotila

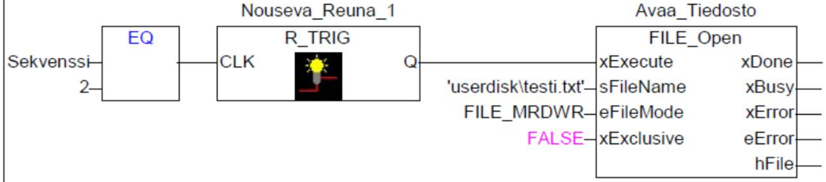
```
VAR_GLOBAL
	Muisti: ARRAY [0..50] OF Taulukko;
	Otsikot: STRING(61);
	Arvot: STRING(255);
	Sekvenssi: WORD;
	Kello: RTC;
	Kolmas_lohko: WORD;
END_VAR
```

```
TYPE Taulukko :
STRUCT
      Aika: DT;
      Nopeus: INT:= 0;
      Taajuus: INT:= 0;
      Virta: INT:= 0;
      Momentti: INT:= 0;
      Teho: INT:= 0;
      DC_Jannite: INT:= 0;
      Lampotila: INT:= 0;
END_STRUCT
END_TYPE
```

```
PROGRAM TXT_Tiedoston_Luonti
VAR
    Nouseva_Reuna_1: R_TRIG;
    Avaa_Tiedosto: FILE_Open;
    Kirjoita_Otsikot: FILE_Write;
    Kirjoituksen_vaihto: FILE_Flush;
    Kirjoita_Arvot: FILE_Write;
    nimi: CAA_FILENAME;
    Sulje: FILE_Close;
    Error_3: BOOL;
    X3: WORD;
    Error_Num_3: WORD;
    Error_4: BOOL;
    X4: WORD;
    Error_Num_4: WORD;
    Error_5: BOOL;
    X5: WORD;
    Error_Num_5: WORD;
    Error_6: BOOL;
    X6: WORD;
    Error_Num_6: WORD;
END_VAR
```

0001

Open file to userdisk



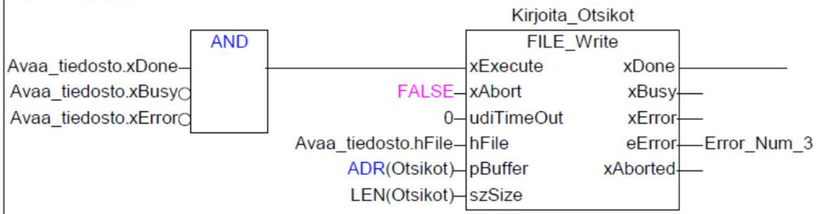
0002

Write headlines to variable Otsikot

'Aika;DC_Jannite;Lampotila;Momentti;Nopeus;Virta;Taajuus;Teho;'———Otsikot

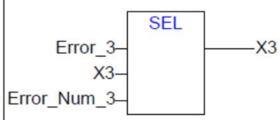
0003

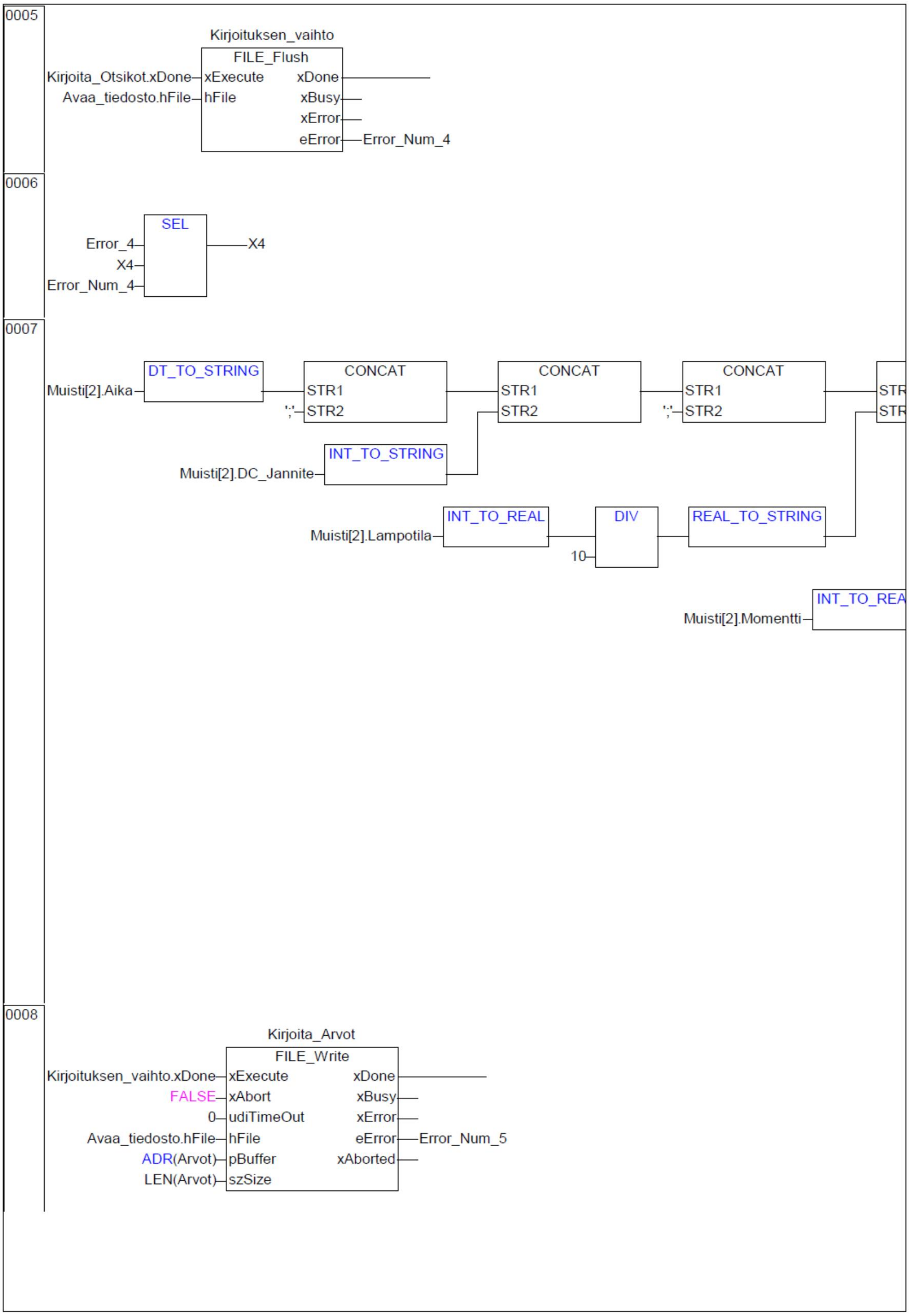
Write variable Otsikot to file

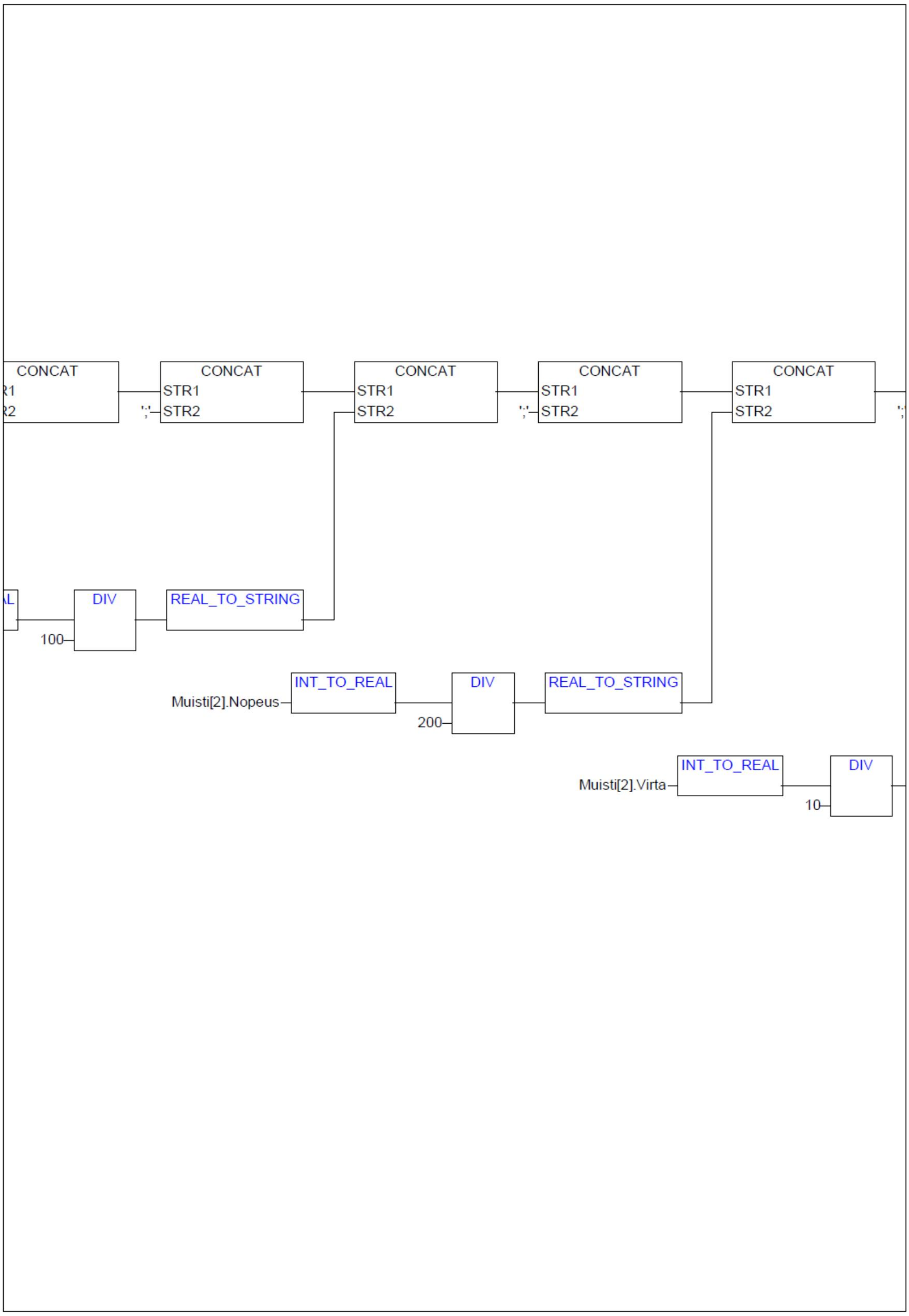


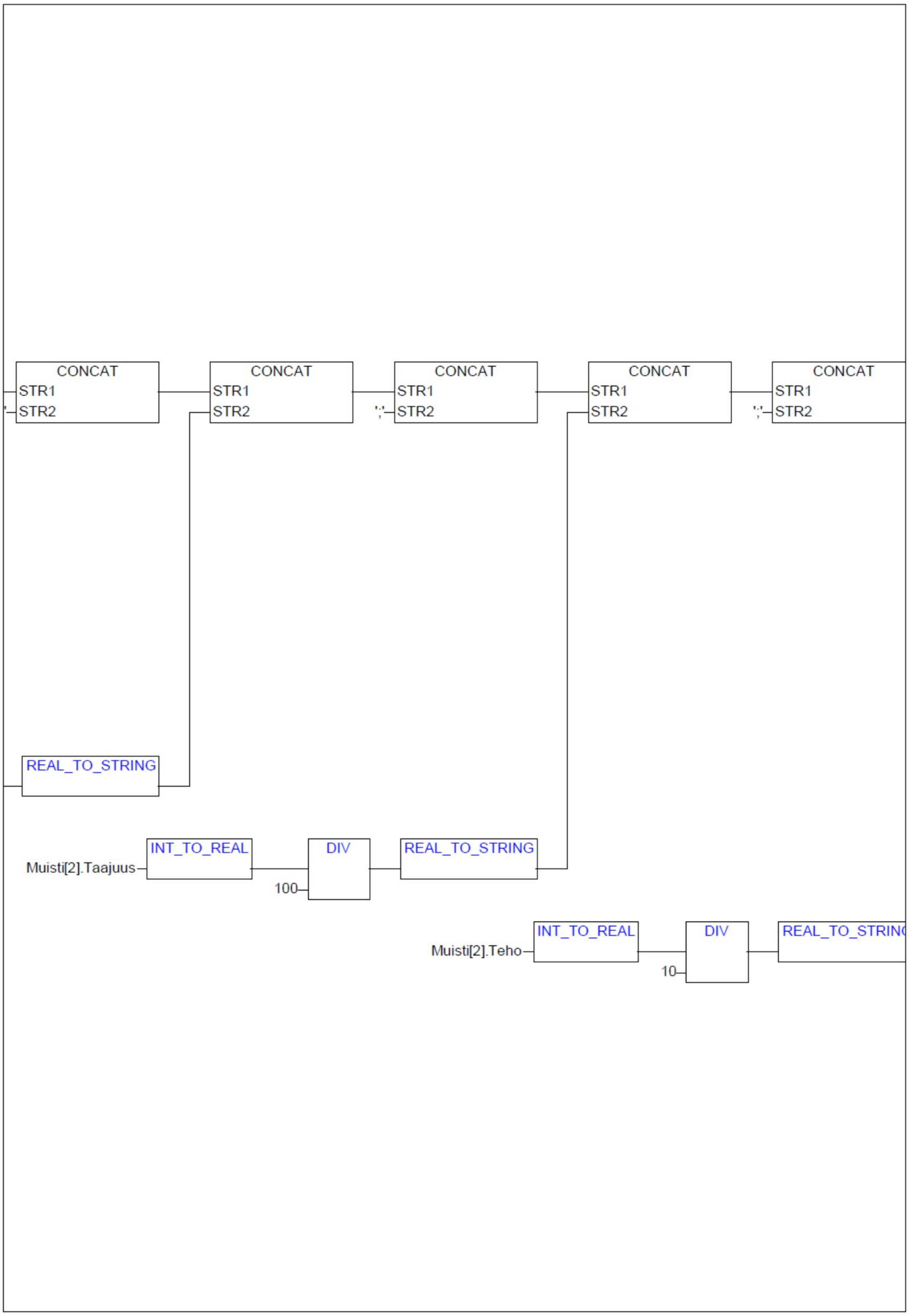
0004

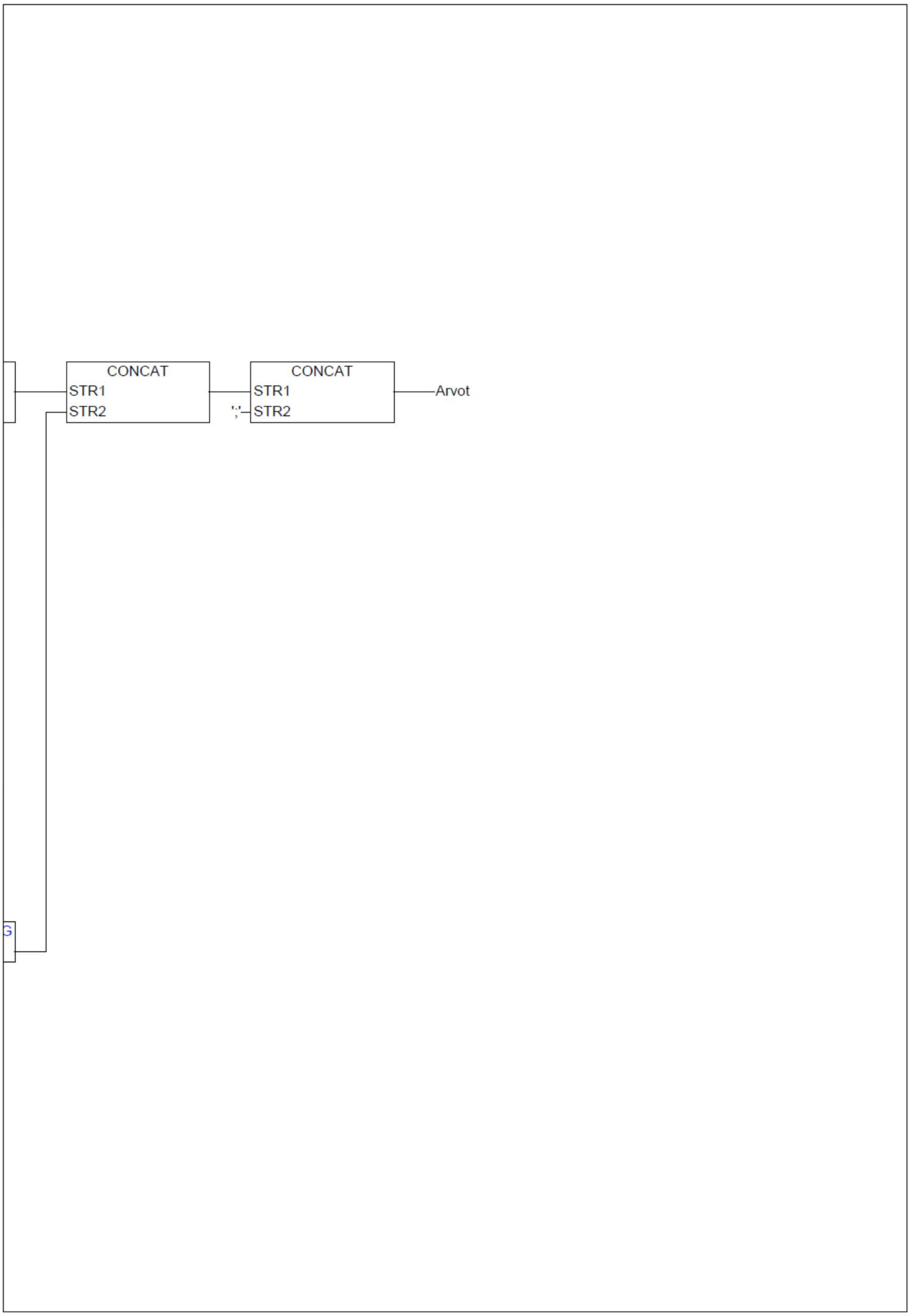
Kirjoita_Otsikot Error check

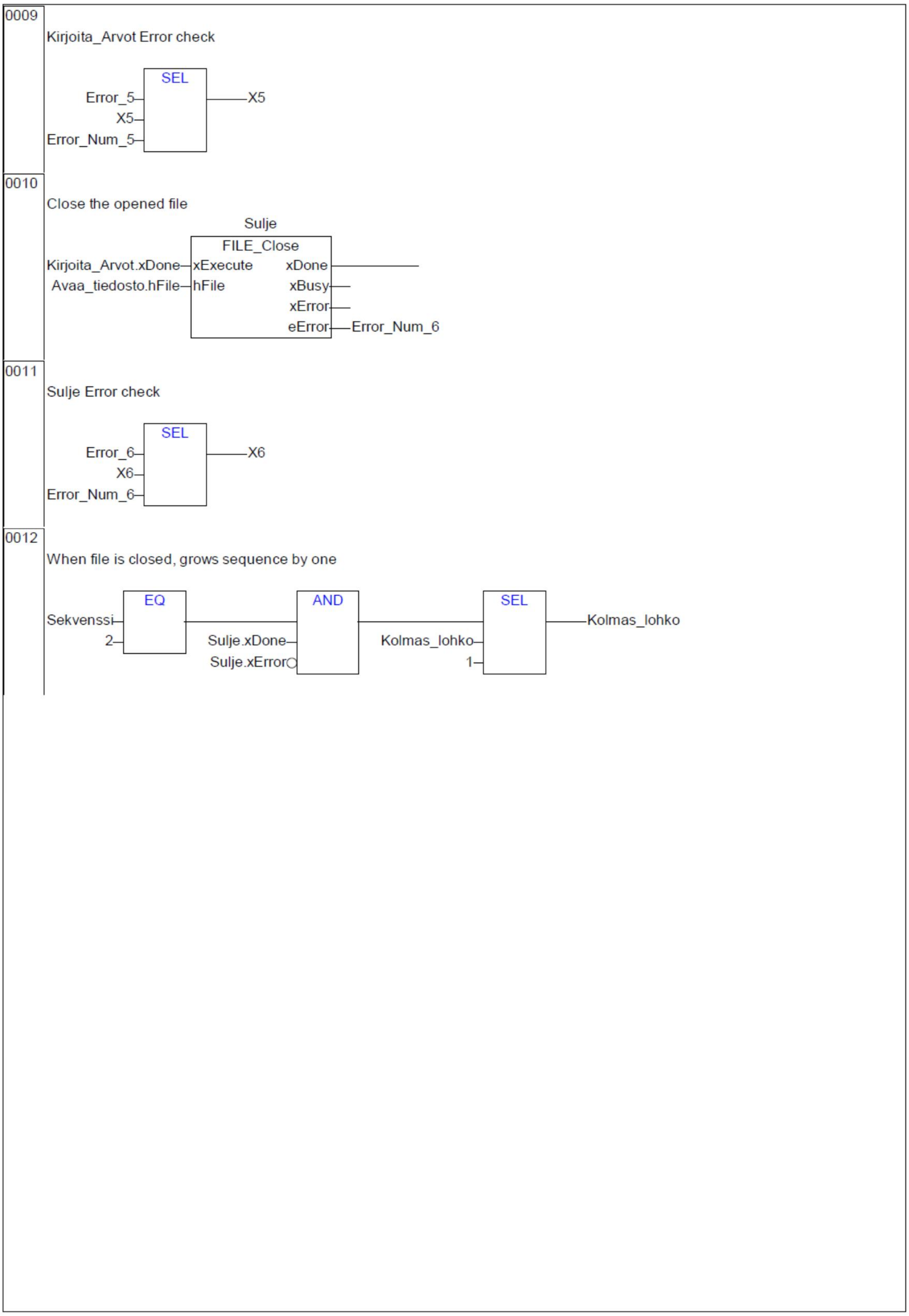



0005
Kirjoituksen_vaihto
FILE_Flush
Kirjoita_Otsikot.xDone
xExecute
xDone
Avaa_tiedosto.hFile
hFile
xBusy
xError
eError
Error_Num_4
0006
SEL
Error_4
X4
Error_Num_4
X4
0007
Muisti[2].Aika
DT_TO_STRING
CONCAT
STR1
';'
STR2
CONCAT
STR1
STR2
CONCAT
STR1
';'
STR2
STR
STR
INT_TO_STRING
Muisti[2].DC_Jannite
INT_TO_REAL
Muisti[2].Lampotila
DIV
10
REAL_TO_STRING
INT_TO_REA
Muisti[2].Momentti
0008
Kirjoita_Arvot
FILE_Write
Kirjoituksen_vaihto.xDone
xExecute
xDone
FALSE
xAbort
xBusy
0
udiTimeOut
xError
Avaa_tiedosto.hFile
hFile
eError
Error_Num_5
ADR(Arvot)
pBuffer
xAborted
LEN(Arvot)
szSize


CONCAT
CONCAT
STR1
';'
STR2
CONCAT
STR1
STR2
CONCAT
STR1
';'
STR2
CONCAT
STR1
STR2
';'
DIV
100
REAL_TO_STRING
Muisti[2].Nopeus
INT_TO_REAL
DIV
200
REAL_TO_STRING
Muisti[2].Virta
INT_TO_REAL
DIV
10


CONCAT
STR1
STR2
CONCAT
STR1
STR2
CONCAT
STR1
';'
STR2
CONCAT
STR1
STR2
CONCAT
STR1
';'
STR2
REAL_TO_STRING
Muisti[2].Taajuus
INT_TO_REAL
DIV
100
REAL_TO_STRING
Muisti[2].Teho
INT_TO_REAL
DIV
10
REAL_TO_STRIN


CONCAT
STR1
STR2
CONCAT
STR1
';'
STR2
Arvot
G


0009
Kirjoita_Arvot Error check
SEL
Error_5
X5
Error_Num_5
X5
0010
Close the opened file
Sulje
FILE_Close
Kirjoita_Arvot.xDone
Avaa_tiedosto.hFile
xExecute
hFile
xDone
xBusy
xError
eError
Error_Num_6
0011
Sulje Error check
SEL
Error_6
X6
Error_Num_6
X6
0012
When file is closed, grows sequence by one
EQ
Sekvenssi
2
AND
Sulje.xDone
Sulje.xError
SEL
Kolmas_lohko
1
Kolmas_lohko